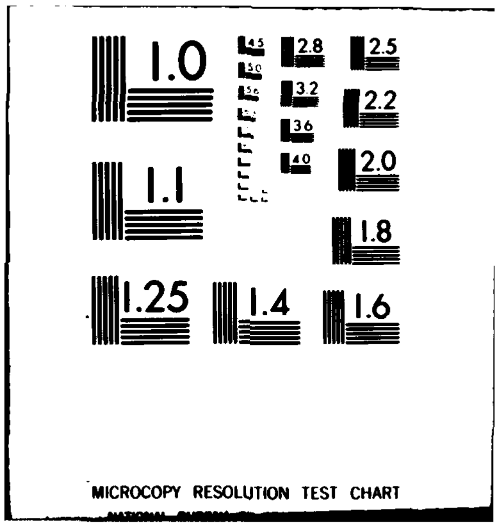
AD-A087 423 LOCKHEED MISSILES AND SPACE CO INC PALO ALTO CA PALO --ETC F/G 13/13
STINT/CD: A STAND-ALONE EXPLICIT TIME INTEGRATION PACKAGE FOR S--ETC(U)
JUN 80 P G UNDERWOOD, K C PARK N00014-74-C-0355

UNCLASSIFIED LMSC/D770637 NL

1 of 1
AD
A087423

END
DATE
FILMED
9-80
DTIC



MICROCOPY RESOLUTION TEST CHART

LEVEL II

(12)

ADA087423

STINT/CD: A STAND-ALONE EXPLICIT TIME INTEGRATION PACKAGE

FOR STRUCTURAL DYNAMICS ANALYSIS

Philip Underwood and K. C. Park
Applied Mechanics Laboratory
Department 5233, Building 205
Lockheed Palo Alto Research Laboratory
3251 Hanover Street
Palo Alto, California 94304

June 1980

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER 14 LMSC-D770637	2. GOVT ACCESSION NO. AD-A087423	3. RECIPIENT'S CATALOG NUMBER 9
4. TITLE (and Subtitle) 6 STINT/CD: A STAND-ALONE EXPLICIT TIME INTEGRATION PACKAGE FOR STRUCTURAL DYNAMICS ANALYSIS.		5. TYPE OF REPORT & PERIOD COVERED Technical Report.
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) 10 Phillip ~~P.~~ G. Underwood [redacted] K. C. Park		8. CONTRACT OR GRANT NUMBER(s) 15 N00014-74-C-0355
9. PERFORMING ORGANIZATION NAME AND ADDRESS Applied Mechanics Laboratory (52-33/205) LOCKHEED PALO ALTO RESEARCH LABORATORY 3251 Hanover Street, Palo Alto, CA 94304		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 12 45
11. CONTROLLING OFFICE NAME AND ADDRESS Office of Naval Research Department of the Navy Arlington, VA 22217		12. REPORT DATE 11 Jun[redacted]80
		13. NUMBER OF PAGES 44
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for Public Release; Distribution Unlimited.

Accession For
NTIS
DDC TAB
By

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

A

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Direct time integration technique, central difference method, variable time step strategy, automatic error control, modular computer implementation.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This paper is a user's guide for the stand-alone explicit direct time integration package STINT/CD for structural dynamics analysis. STINT/CD uses an automatic variable time increment central difference method. The purpose, function, limitations, and usage of the package are described. A FORTRAN listing of STINT/CD is given along with a sample problem which illustrates its usage and performance.

DD FORM 1473 1 JAN 73 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

210118

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

CENTRAL DIFFERENCE TIME INTEGRATOR

Abstract

This paper is a user's guide for the stand-alone explicit direct time integration package STINT/CD for structural dynamics analysis. STINT/CD uses an automatic variable time increment central difference method. The purpose, function, limitations, and usage of the package are described. A FORTRAN listing of STINT/CD is given along with a sample problem which illustrates its usage and performance.

1. Introduction

Direct time integration of the discrete equations of motion governing linear and nonlinear structural dynamics is a frequently used solution method for transient response analysis. It would appear that the central difference integrator is the most commonly used explicit method; for example, it is used in the HONDO [1] and STAGS [2] computer codes. A difficulty with the central difference integrator has been in the selection of the time increment to maintain stability and desired accuracy over a range of structural behavior and loading conditions. A central difference integration method [3,4] has recently been developed to overcome this difficulty. This central difference method [3,4] automatically selects the time increment to maintain stability and to achieve user requested accuracy. The time increment is selected so that user specified sampling rates with respect to the dominant and maximum "apparent" response frequencies are maintained. In addition the central difference method [3,4] is implemented as a stand-alone package (STINT/CD) so that it is easily interfaced with existing structural analyzers (finite -element and -difference computer codes).

This paper is a user's guide for STINT/CD (Stand-alone Time INTegrator/Central Difference) and includes a FORTRAN listing (Appendix A). The user's guide presents: 1) a description of this automatic variable time increment central difference method, 2) a description of the user written subroutines (examples are listed in Appendix B) required to interface with a host structural analyzer, 3) a sample problem (embedded in the example user written subroutines) which illustrates many of the features of this method, and 4) some suggestions for usage of STINT/CD. The subroutines are written in FORTRAN IV, utilize in-core storage after an initial data transfer from mass storage, and are operational with minor changes on the DEC VAX-11/780, UNIVAC 1100, and CDC 6000/7000 computer operating systems.

2. Method Description

The purpose and function of this central difference method are described briefly; to the extent that a user can apply it to a specific problem. The theory and implementation underlying the method are contained in references [3,4]; the user is encouraged to read these references before using the method. In the function description the error measures are reviewed because there are additions since publication of references [3,4]. This section concludes with a description of the limitations of the method.

Purpose

The time integrator package STINT/CD, which is comprised of the subroutines STINTC, CENDIF, ACLTN, ERRORE, STEPSZ, ROTATE, and VIPDA given in Appendix A, solves the discrete equation of motion

$$\underset{\sim}{M}\, \ddot{\underline{u}}^n + \underset{\sim}{D}\, \dot{\underline{u}}^n + \underline{f}_s(\underline{u}^n) = \underline{f}(t^n) \tag{1}$$

governing structural dynamics, where $\underline{u}$ is the computational displacement vector, the superscript n indicates the vector value at the discrete time t^n, the superscript dot · denotes temporal differentiation, $\underset{\sim}{M}$ is a diagonal mass matrix, $\underset{\sim}{D}$ is a damping matrix, $\underline{f}_s(\underline{u}^n)$ is the set of internal forces due to the stiffness that oppose the structural deformation (stiffness forces), and $\underline{f}(t^n)$ is the applied load. For a linear problem the stiffness force $\underline{f}_s(\underline{u})$ becomes $\underset{\sim}{K}\, \underline{u}$, where $\underset{\sim}{K}$ is the linear stiffness matrix. The damping term is treated as $\underline{f}_d = \underset{\sim}{D}\, \dot{\underline{u}}$, or $\underset{\sim}{D}$ itself may be used if it is diagonal. Therefore all quantities in eqn. (1) are computational vectors; i.e. one dimensional arrays of length equal to the number of degrees of freedom.

Function

The subroutine STINTC provides an internal control interface between the host structural analyzer and the subroutine CENDIF which is the "main" subroutine for the variable time increment central difference integrator. The input to STINTC is supplied by the user through the user written subroutine DRIVER which is described in Section 3. In addition the diagonal $\underset{\sim}{M}$ matrix (a vector containing the diagonal), the $\underset{\sim}{D}$ matrix, if diagonal and the matrix option is chosen, and the initial conditions $\underline{u}^{o}$ and $\underline{\dot{u}}^{o}$, if present, are read from mass storage.

The subroutine CENDIF, called by STINTC, contains the integrator formulas for the fixed time increment and increasing or decreasing time increments; see [3,4] for the details of the various formulas. Both $\underline{u}^{n}$ and $\underline{\dot{u}}^{n-\frac{1}{2}}$ are computed based on the acceleration at t^{n-1}. The acceleration is computed in subroutine ACTLN for the cases: 1) no damping, 2) diagonal damping, and 3) nondiagonal damping; again, the details can be found in references [3,4]. Before the new displacement $\underline{u}^{n}$ and the velocity $\underline{\dot{u}}^{n-\frac{1}{2}}$ are accepted, the error for this step is computed in subroutine ERRORE which is discussed below. Based on the error computation the time increment for the next step is computed in subroutine STEPSZ; see [4] for the details. Once the step is accepted the computed displacement $\underline{u}^{n}$ and velocity $\underline{\dot{u}}^{n}$ (extrapolated to the same time as the displacement) and t^{n} are transferred through the user written subroutine OUTPUT for display by printing, plotting, etc.

In subroutine ERRORE the "maximum perturbed apparent" frequency [3,4] and the "dominant apparent frequency" (a new frequency measure, discussed below) error measures are computed. From [4] the "maximum perturbed apparent frequency" error measure ε_m is

$$\varepsilon_m = \max(\varepsilon_1^n, \ldots, \varepsilon_{MAXDEG}^n) \qquad (2)$$

where,

$$\varepsilon_i^n = (h_n^2/4)(a_i^n/b_i^n)$$

$$a_i^n = |\ \ddot{u}_i^n - \ddot{u}_i^{n-1}\ |$$

$$b_i^n = \max_{|i-j| \le m_b} (|\ u_j^n - u_j^{n-1}\ |)$$

and MAXDEG is the size (dimension) of the solution vector in eqn. (1), h_n is the n-th time increment, and m_b is the bandwidth (average) of the i-th degree of freedom. The "dominant apparent frequency" error measure ε_d is

$$\varepsilon_d = \frac{h_n^2}{4}\sqrt{\frac{(\Delta\ \underline{u}^n)^T\ \underset{\sim}{M}\ \Delta\ \underline{\ddot{u}}_n}{(\Delta\ \underline{u}^n)^T\ \underset{\sim}{M}\ \Delta\ \underline{u}^n}} \qquad (3)$$

where

$$\Delta \underline{u}^n = \underline{u}^n - \underline{u}^{n-1}$$

$$\Delta \ddot{\underline{u}}^n = \ddot{\underline{u}}^n - \ddot{\underline{u}}^{n-1}$$

and superscript T indicates transposition. The argument of the square root in eqn. (3) is recognized as a form of Rayliegh's quotient. Hence it gives an estimate of the frequency of the global (dominant) change in the solution vector $\underline{u}^n$ at t^n . This measure has been found to accurately estimate the dominant frequency of the response. The user controls the stability and accuracy by specifying the number of samples/cycle for both the maximum and dominant apparent frequencies; see [3,4] for the relationship between samples/cycle and the error measures given by eqns. (2) and (3). Suggested values for the samples/cycle are given in the listing for the user written subroutine DRIVER and the integrator performance for various samples/cycles is presented with the sample problem in Section 4.

The two other subroutines ROTATE and VIPDA are utility subroutines that update vector address pointers and compute the vector inner product, respectively.

The subroutines listed in the Appendices are the DEC VAX-11/780 versions. The only changes required to use these subroutines on a UNIVAC or CDC computer are: 1) the INCLUDE command for inserting the labelled common CENTDF into the subroutines, and 2) the inline comment symbol ! in subroutine STEPSZ and the user written subroutines (Appendix B).

Limitations

A fixed or variable time increment may be selected. For the fixed time increment no check is made on the accuracy or stability. For the variable time increment the user specifies the minimum and maximum time increment and samples/cycle for the maximum and dominant response frequencies. It is recommended that the minimum time increment be a conservative estimate (very small), because the automatic time increment strategy will work more effectively with little loss in efficiency. If more runs are made on the same or similar problems, the minimum time increment can be increased if the performance of the previous run indicates the initial choice was too conservative.

This algorithm will constrain a degree of freedom to a zero value if the corresponding mass matrix entry is zero. No other constraints are allowed.

The diagonal damping option has received only minimal usage so users should study results from this option before accepting them. The no damping and nondiagonal damping options have been exercised for many problems and the authors have confidence in their performance.

If the algorithm is used in the variable time increment mode for pure wave propagation response (i.e. integration of the linear wave equation), it will generally not perform at its best. In some cases the results are terrible; see [4] for an example. Note that, the performance for this problem with a properly chosen fixed time increment is excellent.

3. User Written Subroutines

The user written subroutines are: DRIVER, FORCE, DFORCE, SFORCE, and OUTPUT; they are shown schemetically in Figures 1 and 2. First, DRIVER transmits the problem parameters to the time integrator through the arrays INTGR, LOGIC, and REALN, plus the starting address in core that is to be used by the time integrator. Second, the applied load subroutine FORCE, provides the load vector data to the time integrator. Third, subroutine DFORCE provides the damping force, and fourth, subroutine SFORCE provides the stiffness force. Finally, subroutine OUTPUT provides display of the response. Note that, all data is transmitted as vectors, inasmuch as diagonal matrices may be considered vectors.

The listing of the user written subroutines in Appendix B includes the purpose and usage requirements of each subroutine. In this case the DRIVER subroutine also includes the mass matrix and initial velocity data for the example problem: normally this data is generated in the host structural analyzer. Here the structural analyzer is embedded in the user written interface subroutines and the DRIVER subroutine is the main program. In a typical application the user written subroutine DRIVER is called by the host structural analyzer and the user written subroutines FORCE, DFORCE, SFORCE, and OUTPUT call the host structural analyzer to furnish the required data.

4. Sample Problem

The sample problem is the two degree of freedom nonlinear model shown in Figure 3. For this model the kinetic energy, T, potential energy, V, and the dissipation function, F, are given by

$$T = \tfrac{1}{2} m_1 \dot{u}_1^2 + \tfrac{1}{2} m_2 \dot{u}_2^2$$

$$V = k_1 \log(\cosh u_1) + \tfrac{1}{2} k_c (u_1 - u_2)^2 + k_2 \cosh u_2 \qquad (4)$$

$$F = \tfrac{1}{2} d\, (\dot{u}_1 - \dot{u}_2)$$

From Lagrange's equation [5],

$$\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{u}_i}\right) - \frac{\partial T}{\partial u_i} + \frac{\partial F}{\partial \dot{u}_i} + \frac{\partial V}{\partial u_i} = P_i \qquad (5)$$

the mass matrix, damping forces, and stiffness forces are found to be

$$\underset{\sim}{M} = \begin{bmatrix} m_1 & 0 \\ 0 & m_2 \end{bmatrix}$$

$$d_1 = d(\dot{u}_1 - \dot{u}_2)$$
$$d_2 = d(\dot{u}_2 - \dot{u}_1) \tag{6}$$

$$f_1 = k_1 \tanh u_1 + k_c (u_1 - u_2)$$
$$f_2 = k_2 \cosh u_2 + k_c (u_2 - u_1)$$

where $m_1 = m_2 = 1.0$, $d = 5.0$, $k_1 = k_2 = 1000.0$ and $k_c = 100.0$. The initial conditions and the forcing function are shown in Figure 3.

Physically, the system comprises a softening element with an initial velocity and a hardening element subjected to a time delayed rectangular force with the two nonlinear elements moderately coupled by a linear spring and dashpot. This model was chosen because the response illustrates how well the time increment selection strategy of the time integrator package works.

The printed output for the sample problem is listed in Appendix C. The first three lines, the time increment data and the synopsis data are printed from the STINT/CD package. The response data are printed in subroutine OUTPUT. The example problem was also run for 20, 100, and 500 samples/cycle on the dominant frequency (REALN(5)) in addition to the 50 samples/cycles run shown in Appendix C. All other problem parameters are held constant. In Figure 4 the time increment versus the response time is shown. In general the time increment is increased at the beginning followed by decreases after $t = 0.5$ sec., when the rectangular load is applied to the hardening element. This behavior is very reasonable. A large time increment is possible for $t < 0.5$ sec. because the softening element dominates the response, but later the hardening element, with a higher frequency, dominates the response. Also, note that as the sampling rate is increased (more accuracy) the time increment consistently decreases.

To illustrate that convergence is obtained the example problem was also run for a fixed time increment of 0.005 sec. This time increment is much smaller than any time increment selected by the time integrator algorithm, so it should be sufficiently small to provide a converged solution. The displacement and velocity history for the four variable time increment runs and the fixed time increment run are shown in Figures 5-8. During the first half of the response the results are nearly identical, but afterward some slight differences are seen. Note that the response for the 500 samples/cycle and the fixed time increment are identical; look near the very end of the response histories for the clearest picture.

In a fixed time increment mode this problem will remain stable for a time increment up to approximately 0.03 sec. At this time increment though the accuracy is minimal. For a non-stiff (closely spaced eigenvalues) problem with a small (1-3) number of degrees of freedom accuracy considerations usually dominate over stability considerations. For a stiff (widely separated eigenvalues) problem with more degrees of freedom, such as the cantilever beam [4], stability dominates and the accuracy is well within what the user requests.

For the variable time increment mode the behavior of the average time increment versus the requested samples/cycle of the dominant apparent frequency, shown in the Table below, sheds more light on stability versus accuracy.

Dominant Frequency Samples/Cycle	Average Time Increment (sec.)
20	0.0062112
50	0.0056818
100	0.0045455
500	0.0010917

Only the time increment change from 100 to 500 samples/cycle shows a decrease in proportion to the samples/cycle ratio. At the other sampling rates the time increment is being controlled by a mixture of stability and accuracy. Hence the ratios between the sampling rate and the average time increment do not correlate unless accuracy clearly controls the time increment.

5. Suggested Usage

The stand-alone in-core time integrator package STINT/CD presented here is easily adapted to problems with several hundred degrees of freedom. However, before attempting a problem this large, it is suggested that the user experiments with a small problem with characteristics similar to a larger problem that one is interested in solving. The user may also want to change some of the algorithm parameters in the subroutine STEPSZ to fit a particular class of problems more efficiently. References [3,4] should be studied before adjusting the time increment selection parameters.

In the authors' work environment (solving a variety of interdisciplinary problems) the stand-alone feature is most desireable, since it allows the quick assembly of software elements to solve the current problem. However, in a strictly production environment, where the mechanics of the problem seldom change, some efficiency may be gained by embedding the time integrator into the structural analyzer; see [6].

The explicit central difference integrator is most suited to short duration transient loads or problems in which the stiffness may suddenly change. The automatic variable time increment feature is especially suited to these problems in that it selects the largest possible time increment consistent with maintaining stability and accuracy based on the current system behavior. Therefore a small time increment does not have to be used during the entire computations to achieve accuracy during one small time interval, since the small time increment is automatically selected only when it is needed.

6. Acknowledgement

The authors wish to acknowledge support of this work by the Office of Naval Research under Contract NOOO14-74-C-0355 and by the Lockheed Missiles & Space Company's Independent Research Program.

7. References

1. Key, S. W., Beisinger, Z. E. and Krieg, R. D., "HONDO II A Finite Element Computer Program for the Large Deformation Dynamic Response of Axisymmetric Solids", SAND78-0422, Sandia Laboratories, Albuquerque, NM 87185 (October 1978)

2. Almroth, B. O. and Brogan, F. A., "The STAGS Computer Code", NASA CR 2950, National Aeronautics and Space Administration, Washington, DC 20546 (1978)

3. Park, K.C. and Underwood, P.G., "A Variable-Step Central Difference Method for Structural Dynamics Analysis, Part I - Theoretical Aspects", to appear Computer Methods in Applied Mechanics and Engineering (1980)

4. Underwood, P.G. and Park, K.C., "A Variable-Step Central Difference Method for Structural Dynamics Analysis, Part II - Implementation and Performance Evaluation", to appear Computer Methods in Applied Mechanics and Engineering (1980)

5. Meirovitch, L., <u>Analytical Methods in Vibrations</u>, The MacMillan Company, Toronto, (1967)

6. Felippa, C. A. and Park, K. C., "Direct Time Integration Methods in Nonlinear Structural Dynamics", <u>Computer Methods in Applied Mechanics and Engineering</u> 17/18 277-313 (1979)

Appendix A

STINT/CD Computer Code Listing

```
      SUBROUTINE STINTC(LOGIC,INTGR,REALN,C)
C
C *** PURPOSE      PROVIDES THE INTERFACE BETWEEN THE USER SUPPLIED
C                  'DRIVER' ROUTINE AND THE CENTRAL DIFFERENCE TIME
C                  INTEGRATOR CENDIF/VECTOR (CALL CENDIF)
C
C *** INPUT        LOGIC  ARRAY OF LOGICAL CONTROL DATA
C                  INTGR  ARRAY OF INTEGER CONTROL DATA
C                  REALN  ARRAY OF REAL CONTROL DATA
C                  C  STARTING ADDRESS OF COMMON (CORE) TO BE USED BY
C                     THE INTEGRATOR
C
C
      EXTERNAL GASPER
      INCLUDE 'PRODCD.PCR'
C
C
      LOGICAL LOGIC(1)
      LOGICAL IGASP,IDISP,IVEL
      INTEGER INTGR(1)
      REAL    C(1)
      REAL    REALN(1)
C
      IGASP  = LOGIC(1)
      IFORCE = LOGIC(2)
      IDISP  = LOGIC(3)
      IVEL   = LOGIC(4)
      FIXSTP = LOGIC(5)
      IDAMP  = LOGIC(6)
      IDMPDG = LOGIC(7)
C
      MAXDEG = INTGR(1)
      IUNIT  = INTGR(2)
      KBAND  = INTGR(3)
      KBAND = KBAND - 1

C
      TIME   = REALN(1)
      TMAX   = REALN(2)
```

```
      DTMIN  = REALN(3)
      DTMAX  = REALN(4)
      EPSDFQ = REALN(5)
      EPSHFQ = REALN(6)
      ALFA   = REALN(7)
      BTA = 0.5*ALFA
C
      IF(FIXSTP) DTMIN = DTMAX
      DT = DTMIN
      MAXSTP = 3*TMAX/DTMIN
      NSTEP = 1
C
      IF(EPSHFQ .LT. 3.14) EPSHFQ = 4.0
      EPSHFQ = (3.14159265/EPSHFQ)**2
      IF(EPSDFQ .LT. 3.14) EPSDFQ = 4.0
      EPSDFQ = (3.14159265/EPSDFQ)**2
      UPNTR(1) = 0
      UPNTR(2) = 1
      UPNTR(3) = 2
      MAXVEC = 3
      IF(FIXSTP) MAXVEC = 1
      IF(FIXSTP) UPNTR(2) = 0
      IF(FIXSTP) UPNTR(3) = 0
      NDOUBL = 0
      NCUTS = 0
      NFACTS = 0
C
C *** STORAGE ALLOCATION
C
      M = MAXDEG
      MM = MAXVEC*MAXDEG
C
      LSM = 1
      LRSM = LSM + M
      IF(FIXSTP) LRSM = LSM
      LSC = LRSM + M
      LU = LSC
      IF(IDMPDG) LU = LSC + M
      LUD = LU + MM
      LUDD = LUD + MM
      LW = LUDD + MM
      LWORKA = LW
      IF((IDMPDG) .OR. (IDAMP)) LWORKA = LW + M
      LWORKB = LWORKA + M
      LEND = LWORKB + M - 1
C
      WRITE(6,200) LEND
  200 FORMAT(1H1, 2X,'WELCOME TO STINTC THE STAND-ALONE CENTRAL DIFFEREN
     1CE TIME INTEGRATOR',//,17X,'STINTC REQUIRES',I10,'  WORDS OF STORA
     2GE',//,27X,'6 DECEMBER 1979  VERSION',//)
C
      DO 10 I=1,LEND
         C(I) = 0.0
   10 CONTINUE
C
```

```
      IF(IGASP)                                GO TO 20
C *** UNFORMATTED READS
      READ(IUNIT) (C(LSM-1+I),I=1,M)
      IF((IDMPDG) .AND. (IDAMP)) READ(IUNIT) (C(LSC-1+I),I=1,M)
      IF(IDISP) READ(IUNIT) (C(LU-1+I),I=1,M)
      IF(IVEL) READ(IUNIT) (C(LUD-1+I),I=1,M)
      CLOSE(UNIT=IUNIT)
                                               GO TO 30
C
C *** DMGASP READS  NOTE: IF NOT AVAILABLE REPLACE BY OTHER EFFICIENT
C ***                     I/O PACKAGE
   20 CALL DM RAST (IUNIT, C(LSM), M)
      CALL ETS PRT(1,'STINTCR1')
      CALL DM HAST (0, GASPER, 0)
      IF((IDMPDG) .AND. (IDAMP)) CALL DM RAST (IUNIT, C(LSC), M)
      CALL ETS PRT(1,'STINTCR2')
      CALL DM HAST (0, GASPER, 0)
      IF(IDISP) CALL DM RAST (IUNIT, C(LU), M)
      CALL ETS PRT(1,'STINTCR3')
      CALL DM HAST (0, GASPER, 0)
      IF(IVEL) CALL DM RAST (IUNIT, C(LUD), M)
      CALL ETS PRT(1,'STINTCR4')
      CALL DM HAST (0, GASPER, 0)
      CALL DM FAST (IUNIT, 0, 0)
   30 CONTINUE
C
      CALL CENDIF( C(LSM), C(LRSM),  C(LSC), C(LU), C(LUD), C(LUDD),
     1             C(LW), C(LWORKA), C(LWORKB))
C
      RETURN
      END



      SUBROUTINE CENDIF(SM, RSM, SC, U, UD, UDD, W, WORKA, WORKB)
C
C
C     SOLVES SECOND ORDER STRUCTURAL DYNAMICS BY EXPLICIT CENTRAL
C     DIFFERENCE METHOD
C
C
C *** EQUATIONS OF MOTION
C
C        M*UDD + D*UD + K*U = F
C
C        WHERE  M IS A DIAGONAL MASS MATRIX
C               D IS THE DAMPING MATRIX
C               K*U IS THE FORCE DUE TO STIFFNESS (LINEAR OR NONLINEAR)
C
C *** VECTOR FORMULATION (IE D*UD AND K*U ARE AVAILABLE AS VECTORS)
C
C *** DEFINITIONS
C
C        W = UD + 0.5*ALFA*(HN+HNM1)*(DV - D*UD/M) (NONDIAGONAL
C            DAMPING)
```

```
C         W  USED TO STORE DAMPING FORCE (DIAGONAL DAMPING)
C
C         HN  THE TIME STEP FROM N TO N+HALF
C         HNM1 THE TIME STEP FROM N-HALF TO N
C
C *** STORAGE
C
C         SM  MASS MATRIX (DIAGONAL STORED AS A VECTOR)
C         RSM  RECIPROCAL OF MASS MATRIX
C         SC  DAMPING MATRIX (ONLY REQUIRED FOR DIAGONAL DAMPING, STORED
C             AS A VECTOR)
C         U (3 MOST CURRENT VALUES FOR VARIABLE STEP)
C         UD (3 MOST CURRENT VALUES FOR VARIABLE STEP)
C         UDD (3 MOST CURRENT VALUES FOR VARIABLE STEP)
C         W  SEE DEFINITION ABOVE
C         SCRATCH VECTORS WORKA,WORKB
C
      INCLUDE 'PRODCD.PCR'
C
C
      INTEGER NTIME(28)
      REAL SM(1)
      REAL RSM(1)
      REAL SC(1)
      REAL U(1),UD(1),UDD(1)
      REAL W(1)
      REAL WORKA(1),WORKB(1)
C
C *** INITIAL VALUES
      NSOLV = 0
      IRST = 0
      ILAST = 0
      IDTMX = 0
      INCTRY = 0
      IDEC = 0
      INCRS = .FALSE.
      DECRS = .FALSE.
      ERR = 0.0
      RN = 1.
C *** INITIALIZE TIME STEP OCCURENCE ARRAY
      DO 10 I=1,28
         NTIME(I) = 0
   10 CONTINUE
C *** FORM INVERSE OF DIAGONAL MASS MATRIX SM
C *** NOTE  IF SM=0.0 DOF IS CONSTRAINED TO 0.0
      DO 20 I=1,MAXDEG
         IF(SM(I) .EQ. 0.0)            GO TO 20
         RSM(I) = 1./SM(I)
   20 CONTINUE
C *** STEP SIZES
   30 CONTINUE
      IF(IRST.EQ.0)  HN = 5.*DTMIN
      IF(HN .GT. (0.5*DTMAX)) HN = 0.5*DTMAX
      IF(FIXSTP) HN = 0.5*DTMAX
      HNM1 = HN
```

```
C
C *** STARTING PROCEDURE
C
C
C *** INITIALIZE WORK SPACE
      DO 100 I=1,MAXDEG
         WORKA(I) = 0.0
         WORKB(I) = 0.0
C     *** W = 1 FOR DIAGONAL DAMPING INITIAL SET UP (IF IDAMP FALSE)
         W(I) = 1.0
  100 CONTINUE
C *** COMPUTE APPLIED FORCE IN WORKA
      IF(IFORCE) CALL FORCE(MAXDEG,TIME,WORKA)
C *** COMPUTE TOTAL SPRING FORCE  (IN WORKB)
      CALL SFORCE(MAXDEG,U,WORKB)
C *** PRINT OUT INITIAL CONDITIONS
      IPRT=2
C     CALL OUTPUT(IPRT,MAXDEG,TIME,U,UD)
      IF(IRST .EQ. 0) CALL OUTPUT(IPRT,MAXDEG,TIME,U,UD)
C *** FOR DIAGONAL DAMPING PUT DAMPING COEFFICIENTS INTO SC
      IF((IDMPDG) .AND. (.NOT. IDAMP)) CALL DFORCE(MAXDEG,W,SC)
C *** COMPUTE DAMPING FORCE (IN W)
      IF((IDAMP) .AND. (.NOT. IDMPDG)) CALL DFORCE(MAXDEG,UD,W)
C *** FORM -F+D*DU+K*U IN WORKA
      DO 110 I=1,MAXDEG
         IF(.NOT.IDAMP) W(I) = 0.0
         IF(IDMPDG) W(I) = SC(I)*UD(I)
         WORKA(I) = -WORKA(I) +WORKB(I) +W(I)
  110 CONTINUE
C
C *** COMPUTE ACCELERATION AT TIME=START, VELOCITY AT TIME=START+HN,
C ***    AND DISPLACEMENT AT TIME=START+2HN
C
      CALL ROTATE(MAXVEC,UPNTR)
      I1 = UPNTR(1)*MAXDEG
      I2 = UPNTR(2)*MAXDEG
      DO 120 I=1,MAXDEG
         J=I1+I
         K=I2+I
         UDD(K) = -RSM(I)*WORKA(I)
         UD(J) = UD(K) + HN*UDD(K)
         U(J) = U(K) + 2.*HN*UD(J)
  120 CONTINUE
C
C *** BEGIN TIME LOOP
C
      NSTEPS = NSTEP-1
  200 NSTEPS = NSTEPS+1
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C
C *** ARRAY POINTERS
      I1 = UPNTR(1)*MAXDEG
      II1 = I1+1
      I2 = UPNTR(2)*MAXDEG
      I3 = UPNTR(3)*MAXDEG
```

```
C *** UPDATE TIME
  300 TIME = TIME + 2.*HN
      NSOLV = NSOLV + 1
      IF(NSTEPS .EQ. NSTEP)               GO TO 340
C
C *** UPDATE VELOCITY AND DISPLACEMENT
C ***    (VELOCITY AT N-HALF, DISPLACEMENT AT N)
C
      DO 330 I=1,MAXDEG
         J=I1+I
         K=I2+I
         L=I3+I
         IF(RN .LE. 1.0)                  GO TO 310
         D = ABS(UDD(K))
         IF(D .EQ. 0.0)                   GO TO 310
         DEM = ABS(UDD(K)-UDD(L))/D
         IF(DEM .GT. 1.E-02)              GO TO 310
C     *** ALMOST CONSTANT ACCELERATION
         UD(J) = UD(K)-0.25*(HN-HNM1)*((3.-RN)*UDD(K)+(1.+RN)*UDD(L))
         UD(J) = UD(J) + 2.0*HN*UDD(K)
                                          GO TO 320
  310    UD(J) = UD(K) + (HN+HNM1)*UDD(K)
  320    U(J) = U(K) + 2.0*HN*UD(J)
  330 CONTINUE
  340 CONTINUE
C
C *** COMPUTE ACCELERATION
C
      CALL ACLTN(RSM,SC,U,UD,UDD,W,WORKA,WORKB)
C
C *** ERROR COMPUTATION
C
      IF(FIXSTP)                          GO TO 430
      CALL ERRORE(SM,U,UDD,ETRNM,ETRDF,WORKA,WORKB)
C
C *** STEP SIZE CONTROL SECTION
C
      CALL STEPSZ(ETRNM,ETRDF,U,UD,UDD,KGO)
      GO TO (400,30,500,300),KGO
C
C *** OUTPUT SECTION
C
  400 CONTINUE
      IDEC = 0
C *** TIME STEP DISTRIBUTION DETERMINATION
      DT = 2.0*HNM1
      TLIM = 0.0
      TFAC = 10.0
      TLIMA = 1.0
      DO 410 INT=1,28
         IF(INT .GT. 10) TFAC = 100.
         IF(INT .GT. 19) TFAC = 1000.
         TLIM = TLIM + TFAC
         IF((DT .GE. TLIMA*DTMIN) .AND. (DT .LT. TLIM*DTMIN))
     1                                    GO TO 420
```

```
            TLIMA = TLIM
   410 CONTINUE
   420 NTIME(INT) = NTIME(INT) + 1
   430 CONTINUE
         IPRT = 0
C *** SET TIME STEP TO END AT TMAX
         IF(TIME .GE. TMAX) ILAST=1
         IF((TIME+2.0*HN) .GT. TMAX) HN = 0.5*(TMAX-TIME)
C *** PRINTING OUTPUT
C *** COMPUTE VELOCITY AT N-STEP FOR PRINTOUT PURPOSES
         DO 440 I=1,MAXDEG
            J=I1+I
            WORKB(I) = UD(J) + HN*UDD(J)
   440 CONTINUE
         IF(ILAST .EQ. 1) IPRT=10
         CALL OUTPUT(IPRT,MAXDEG,TIME,U(II1),WORKB)
         IF(ILAST .EQ. 1)                 GO TO 500
C *** SET POINTERS FOR NEXT TIME STEP
         CALL ROTATE(MAXVEC,UPNTR)
C
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C
C
C *** END TIME LOOP
C
         IF(NSTEPS .LT. MAXSTP)           GO TO 200
C
C *** SUMMARY PRINT OUT OF TIME INTEGRATOR PERFORMANCE
   500 IF(FIXSTP)                         GO TO 520
         TAVE = TIME/(NSTEPS-1)
         WRITE(6,1000)
  1000 FORMAT(1H1)
         WRITE(6,1010) TAVE,NDOUBL,NCUTS,NFACTS,NSTEPS,NSOLV
  1010 FORMAT('  AVERAGE TIME STEP           ',E12.5/,
     12X, 'NUMBER OF STEP INCREASES        ',I5/,
     22X, 'NUMBER OF STEP DECREASES        ',I5/,
     32X, 'NUMBER OF FACTORIZATIONS        ',I5/,
     42X, 'NUMBER OF TIME STEPS            ',I5/,
     52X, 'NUMBER OF SOLUTIONS             ',I5 )
         WRITE(6,1020)
  1020 FORMAT(////,10X,'DT OCCURRENCES IN THE RANGES INDICATED',/)
         TLIM = 0.0
         TFAC = 10.0
         TLIMA = 1.0
         DO 510 INTIM=1,28
            IF(INTIM .GT. 10) TFAC = 100.
            IF(INTIM .GT. 19) TFAC = 1000.
            TLIM = TLIM + TFAC
            NTI = NTIME(INTIM)
            IF(NTI .NE. 0) WRITE(6,1030) TLIMA,TLIM,NTI
  1030      FORMAT('  FROM',F8.1,2X,'TO',F8.1,2X,'TIMES DTMIN, DT OCCURRENC
     1ES WAS',I5)
            TLIMA = TLIM
   510 CONTINUE
C
```

```
  520 RETURN
      END


      SUBROUTINE ACLTN(RSM,SC,U,UD,UDD,W,WORKA,WORKB)
C
C *** PURPOSE      TO COMPUTE THE ACCELERATION AT THE N-TH STEP
C
C *** INPUT        HN  HALF OF CURRENT TIME STEP (COMMON CENTDF)
C                  HNM1  HALF OF PAST TIME STEP (COMMON CENTDF)
C                  TIME  CURRENT TIME (COMMON CENTDF)
C                  RSM  RECIPROCAL MASS MATRIX ARRAY
C                  SC  DIAGONAL DAMPING MATRIX ARRAY
C                  U  DISPLACEMENT ARRAY
C                  UD  VELOCITY ARRAY
C
C *** SCRATCH      W  ARRAY FOR DAMPING
C                  WORKA,WORKB
C
C *** OUTPUT       UDD  ACCELERATION ARRAY
C
      INCLUDE 'PRODCD.PCR'
C
C
      REAL RSM(1),SC(1)
      REAL U(1),UD(1),UDD(1)
      REAL W(1)
      REAL WORKA(1),WORKB(1)
C
      II1 = I1+1
C
C *** INITIALIZE WORKING ARRAYS (PREPARING TO COMPUTE ACCELERATION
C ***    AT N-TH STEP)
      DO 10 I=1,MAXDEG
         WORKA(I) = 0.0
         WORKB(I) = 0.0
   10 CONTINUE
C *** COMPUTE APPLIED FORCE IN WORKA
      IF(IFORCE) CALL FORCE(MAXDEG,TIME,WORKA)
C *** COMPUTE TOTAL SPRING FORCE IN WORKB
      CALL SFORCE(MAXDEG,U(II1),WORKB)
C *** FORM -F+K*U (-WORKA+WORKB) AND FORM DV (IN WORKA)
      DO 20 I=1,MAXDEG
         WORKA(I) = -RSM(I)*(-WORKA(I)+WORKB(I))
   20 CONTINUE
      IF(IDMPDG)                        GO TO 120
      IF(.NOT. IDAMP)                   GO TO 100
C *** COMPUTE DAMPING FORCE IN WORKB
      CALL DFORCE(MAXDEG,UD(II1),WORKB)
C *** FORM W
      DO 30 I=1,MAXDEG
         J=I1+I
         W(I) = RSM(I)*WORKB(I)
         W(I) = UD(J) + BTA*(HN+HNM1)*(WORKA(I)-W(I))
```

```
   30 CONTINUE
C *** COMPUTE DAMPING AS UPDATED
      CALL DFORCE(MAXDEG,W,WORKB)
                                                    GO TO 150
C
C *** COMPUTE ACCELERATION AT N
C
C *** NO DAMPING
  100 DO 110 I=1,MAXDEG
         J=I1+I
         UDD(J) = WORKA(I)
  110 CONTINUE
                                                    GO TO 200
C *** DIAGONAL DAMPING
  120 DO 130 I=1,MAXDEG
         J=I1+I
         F1 = 1. + HN*RSM(I)*SC(I)
         F2 = RSM(I)*SC(I)
         WORKB(I) = WORKA(I)- F2*UD(J)
         WORKB(I) = UD(J)+ HN*WORKB(I)/F1
  130 CONTINUE
C *** DETERMINE ACCELERATION
      CALL DFORCE(MAXDEG,WORKB,W)
      DO 140 I=1,MAXDEG
         J=I1+I
         UDD(J) = WORKA(I) - RSM(I)*W(I)
  140 CONTINUE
                                                    GO TO 200
C *** NONDIAGONAL DAMPING
  150 DO 160 I=1,MAXDEG
         J=I1+I
         UDD(J) = WORKA(I) - RSM(I)*WORKB(I)
  160 CONTINUE
C
  200 RETURN
      END


      SUBROUTINE ERRORE(SM,U,UDD,ETRNM,ETRDF,WORKA,WORKB)
C
C *** PURPOSE      TO COMPUTE THE ERROR BASED ON THE HIGHEST
C                  APPARENT FREQUENCY CONCEPT AND THE
C                  DOMINANT FREQUENCY
C
C *** INPUT        HN  HALF OF CURRENT TIME STEP (COMMON CENTDF)
C                  SM  MASS MATRIX
C                  U  DISPLACEMENT ARRAY
C                  UDD  ACCELEREATION ARRAY
C                  COMMON /CENTDF/
C
C *** OUTPUT       ETRNM  THE ERROR (MAX LOCAL FREQUENCY)
C                  ERTDF  THE ERROR (DOMINANT FREQUENCY)
C
C *** SCRATCH      WORKA,WORKB
```

```
C
      INCLUDE 'PRODCD.PCR'
C
      REAL SM(1),U(1),UDD(1),WORKA(1),WORKB(1)
C
C     APPROPRIATE VALUES OF EPSMAC FOR VARIOUS COMPUTERS ARE
C     CDC 6000/7000 SERIES          7.11E-15 (SINGLE PRECISION)
C     IBM 360/370 SERIES            9.54E-07 (REAL*4 PRECISION)
C     IBM 360/370 SERIES            2.22E-16 (REAL*8 PRECISION)
C     UNIVAC 1108/1110, IBM 7094    1.49E-08 (SINGLE PRECISION)
C
      DATA EPSMAC /9.54E-07/
C
      HT24 = HN*HN
      ETRNM = 0.0
C *** FIND MAXIMUM LOCAL FREQUENCY
      DO 30 I=1,MAXDEG
         J=I1+I
         K=I2+I
         L=I3+I
         WORKA(I) = U(J) - U(K)
         WORKB(I) = SM(I)*(UDD(J) - UDD(K))
         DEM = ABS(U(J)-U(K))
         IF(DEM .EQ. 0.0)                  GO TO 30
         IF((U(J) .NE. 0.0) .AND. (DEM/ABS(U(J)) .LE. EPSMAC))
     1                                     GO TO 30
         IF(KBAND .EQ. 0)                  GO TO 20
         KBEG = I-KBAND
         IF(KBEG .LE. 0) KBEG = 1
         KEND = I+KBAND
         IF(KEND .GT. MAXDEG) KEND = MAXDEG
         DO 10 KBE=KBEG,KEND
            JJ = I1+KBE
            KK = I2+KBE
            D = ABS(U(JJ)-U(KK))
            DEM = AMAX1(DEM,D)
   10    CONTINUE
   20    CONTINUE
         ERR = HT24*(UDD(J)-UDD(K))/DEM
         IF(ABS(ERR).GT.ABS(ETRNM)) ETRNM = ERR
   30 CONTINUE
C *** FIND DOMINANT FREQUENCY
      BOT2 = VIPDA(WORKA,WORKA,MAXDEG)
      TOP = VIPDA(WORKA,WORKB,MAXDEG)
      DO 40 I=1,MAXDEG
         WORKB(I) = SM(I)*WORKA(I)
   40 CONTINUE
      BOT = VIPDA(WORKA,WORKB,MAXDEG)
      IF(BOT2 .EQ. 0.0) BOT = 0.0
      IF(BOT2 .EQ. 0.0) BOT2 = 1.0
      IF((ABS(BOT)/BOT2) .LT. (FLOAT(MAXDEG)*EPSMAC)) BOT = 0.0
      ALAMS = 0.0
      IF(BOT .NE. 0.0) ALAMS = TOP/BOT
      ETRDF = HT24*SQRT(ABS(ALAMS))
C
```

```
   30 CONTINUE
C *** COMPUTE DAMPING AS UPDATED
      CALL DFORCE(MAXDEG,W,WORKB)
                                                        GO TO 150
C
C *** COMPUTE ACCELERATION AT N
C
C *** NO DAMPING
  100 DO 110 I=1,MAXDEG
         J=I1+I
         UDD(J) = WORKA(I)
  110 CONTINUE
                                                        GO TO 200
C *** DIAGONAL DAMPING
  120 DO 130 I=1,MAXDEG
         J=I1+I
         F1 = 1. + HN*RSM(I)*SC(I)
         F2 = RSM(I)*SC(I)
         WORKB(I) = WORKA(I)- F2*UD(J)
         WORKB(I) = UD(J)+ HN*WORKB(I)/F1
  130 CONTINUE
C *** DETERMINE ACCELERATION
      CALL DFORCE(MAXDEG,WORKB,W)
      DO 140 I=1,MAXDEG
         J=I1+I
         UDD(J) = WORKA(I) - RSM(I)*W(I)
  140 CONTINUE
                                                        GO TO 200
C *** NONDIAGONAL DAMPING
  150 DO 160 I=1,MAXDEG
         J=I1+I
         UDD(J) = WORKA(I) - RSM(I)*WORKB(I)
  160 CONTINUE
C
  200 RETURN
      END


      SUBROUTINE ERRORE(SM,U,UDD,ETRNM,ETRDF,WORKA,WORKB)
C
C *** PURPOSE        TO COMPUTE THE ERROR BASED ON THE HIGHEST
C                    APPARENT FREQUENCY CONCEPT AND THE
C                    DOMINANT FREQUENCY
C
C *** INPUT          HN  HALF OF CURRENT TIME STEP (COMMON CENTDF)
C                    SM  MASS MATRIX
C                    U  DISPLACEMENT ARRAY
C                    UDD  ACCELEREATION ARRAY
C                    COMMON /CENTDF/
C
C *** OUTPUT         ETRNM  THE ERROR (MAX LOCAL FREQUENCY)
C                    ERTDF  THE ERROR (DOMINANT FREQUENCY)
C
C *** SCRATCH        WORKA,WORKB
```

```
      RETURN
      END


      SUBROUTINE STEPSZ(ETRNM,ETRDF,U,UD,UDD,KGO)
C
C *** PURPOSE      STEPSIZE SELECTION
C
C *** INPUT        HNM1  HALF OF PAST TIME STEP (COMMON CENTDF)
C                  TIME  CURRENT TIME (COMMON CENTDF)
C                  ETRNM  CURRENT ERROR (HIGH FREQUENCY)
C                  ETRDF  CURRENT ERROR (DOMINANT FREQ)
C                  U  DISPLACEMENT ARRAY
C                  UD  VELOCITY ARRAY
C                  UDD  ACCELERATION ARRAY
C                  COMMON /CENTDF/
C
C *** OUTPUT       HN  HALF OF NEW TIME STEP (COMMON CENTDF)
C                  RN  HN/HNM1 (COMMON CENTDF)
C                  KGO  COMPUTED GOTO CONTROL
C
      INCLUDE 'PRODCD.PCR'
C
      REAL U(1),UD(1),UDD(1)
C
      TERM = ABS(ETRNM)/EPSHFQ
      TERM2 = ABS(ETRDF)/EPSDFQ
      TERM = AMAX1(TERM,TERM2)
C *** POSSIBLE STEP SIZE INCREASE CHECK
      IF(TERM .LT. 0.1)                 GO TO 10
C *** STEP SIZE DECREASE CHECK
      IF(TERM .GT. 1.0)                 GO TO 20
      HNM1 = HN
      RN = 1.0
      INCTRY = 0
      IDEC = 0
      INCRS = .FALSE.
      DECRS = .FALSE.
                                        GO TO 50
C *** STEP SIZE INCREASE
   10 HNM1 = HN
      INCRS = .FALSE.
      INCTRY = INCTRY + 1
      IDEC = 0
      IF(NSTEPS .LE. 6) INCTRY = 0
C *** MUST TRY TO INCREASE FOR 5 CONSECUTIVE STEPS BEFORE INCREASING
      IF(INCTRY.LE.4)                   GO TO 50
      INCTRY = 0
      NDOUBL = NDOUBL + 1
      INCRS = .TRUE.
      DECRS = .FALSE.
      IF(TERM .EQ. 0.0) TERM = 0.001
      RATIO = (1.0/(4.*TERM))**0.25
      RATIO = AMIN1(RATIO,1.5)
```

```
      HN = RATIO*HN
      RN = HN/HNM1
      DTN = 2.*HN
      IF(DTN .GT. DTMAX) HN = 0.5*DTMAX
      IF(DTN .GT. DTMAX) IDTMX = IDTMX+1
      IF(IDTMX .GT. 5)                      GO TO 50
      DTN = 2.*HN
      WRITE(6,200) DTN,TIME
  200 FORMAT('  $$$  STEP SIZE INCREASED TO ',E10.4,'  AT ',E10.4)
                                            GO TO 50
C *** STEP SIZE DECREASE
   20 TIME = TIME - 2.0*HN
      INCTRY = 0
      INCRS = .FALSE.
      DECRS = .TRUE.
      IDTMX = 0
      IDEC = IDEC + 1
C *** IF DECREASE MORE THAN 4 TIMES AT ONE TIME POINT, RESTART
      IF(IDEC .GT. 4)                       GO TO 30
      RATIO = (1.0/(5.*TERM))**0.2
      IF(RATIO .LT. 0.66666667) RATIO = 0.66666667
      IF(RATIO .GT. 0.9) RATIO = 0.9
      HN = RATIO*HN
      RN = HN/HNM1
      DTN = 2.*HN
      NCUTS = NCUTS + 1
      WRITE(6,210) DTN,TIME
  210 FORMAT('  $$$  STEP SIZE DECREASED TO ',E10.4,'  AT ',E10.4)
      IF(DTN .LT. DTMIN)                    GO TO 70
      IF(NSTEPS .NE. NSTEP)                 GO TO 80
C *** FIRST TIME STEP SO WE ARE GOING TO RESTART
      UPNTR(1) = 0
      UPNTR(2) = 1
      UPNTR(3) = 2
      HNM1 = HN
      IRST = 1
      RN = 1.0
      WRITE(6,220)
  220 FORMAT(2X,'$$$',2X,'RESTART')
                                            GO TO 60
   30 CONTINUE
C *** LOAD U AND UD FOR A RESTART AT TIME = TIME
      DO 40 I=1,MAXDEG
         K=I2+I
         U(I) = U(K)
         UD(I) = UD(K) + HNM1*UDD(K)
   40 CONTINUE
C *** INITIALIZE POINTERS
      UPNTR(1) = 0
      UPNTR(2) = 1
      UPNTR(3) = 2
      WRITE(6,230) TIME
  230 FORMAT(2X,'$$$',2X,'RESTART AT TIME = ',E12.5)
      IRST = 1
      IDEC = 0
```

```
      ERR = 0.0
      NSTEP = NSTEPS
      RN = 1.0
      HN = 0.1*HN
      DTN = 2.0*HN
      IF(DTN .LT. DTMIN) HN = 0.5*DTMIN
                                         GO TO 60
C
   50 KGO = 1        ! STEP UNCHANGED OR INCREASED
                                         GO TO 90
   60 KGO = 2        ! RESTART
                                         GO TO 90
   70 KGO = 3        ! DT < DTMIN (ERROR)
                                         GO TO 90
   80 KGO = 4        ! STEP DECREASE
   90 RETURN
      END


      SUBROUTINE ROTATE(N,INDEX)
C
C *** PURPOSE      ROTATE STACK POINTERS
C
C *** INPUT        N  NUMBER OF SUBVECTORS
C                  INDEX  ARRAY OF CURRENT POINTERS
C
C *** OUTPUT       INDEX  NEW POINTERS
C
      INTEGER INDEX(1)
C
      IF(N-2)30,10,20
   10 ITEMP = INDEX(2)
      INDEX(2) = INDEX(1)
      INDEX(1) = ITEMP
      RETURN
   20 ITEMP = INDEX(3)
      INDEX(3) = INDEX(2)
      INDEX(2) = INDEX(1)
      INDEX(1) = ITEMP
C
   30 RETURN
      END


      FUNCTION VIPDA(A, B, N)
C
C *** REAL FUNCTION VIPDA = A DOT B, A AND B OF LENGTH N
C        DOUBLE PRECISION ACCUMULATION
C        RESULT RETURNED IN SINGLE PRECISION
C
      DOUBLE PRECISION ACCUM
      REAL A(1),B(1)
C
```

```
      ACCUM = 0.D0
      DO 10 I=1,N
         ACCUM = ACCUM + DBLE(A(I))*DBLE(B(I))
   10 CONTINUE
      VIPDA = SNGL(ACCUM)
C
      RETURN
      END


C
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C
C     PARAMETERS ASSOCIATED WITH CENTRAL DIFFERENCE INTEGRATOR
C
C - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
C
      COMMON /CENTDF/
     B   BTA,
     D   DECRS,DTMAX,DTMIN,
     E   EPSDFQ,EPSHFQ,ERR,
     F   FIXSTP,
     H   HN,HNM1,
     I   IDEC,IDAMP,IDMPDG,IDTMX,IFORCE,INCRS,INCTRY,IRST,
     2   I1,I2,I3,
     K   KBAND,
     M   MAXDEG,MAXSTP,MAXVEC,
     N   NCUTS,NDOUBL,NFACTS,NSTEP,NSTEPS,
     R   RN,
     T   TIME,TMAX,
     U   UPNTR
C
      INTEGER UPNTR(3)
      LOGICAL DECRS,FIXSTP,IDAMP,IDMPDG,IFORCE,INCRS
C
C
```

Appendix B

User Written Subroutines with Sample Problem

```
C     SUBROUTINE DRIVER      IN THIS EXAMPLE DRIVER IS THE MAIN PROGRA
C
C --- PURPOSE       TO PROVIDE THE DATA NEEDED TO CALL STINTC (ELEMENT
C                   STINT/CENDIF) FOR CENTRAL DIFFERENCE TIME INTEGRATI
C
C --- INPUT         USER SUPPLIED, SEE LIST NEEDED BELOW
C
C --- OUTPUT        THE ARRAYS     LOGIC(20)
C                                  INTGR(20)
```

```
C                                    REALN(20)
C                                    C(ICORE)
C
C        WHERE
C            LOGIC(1) = IGASP, IF .TRUE. DMGASP I/O USED
C                                 .FALSE. UNFORMATTED FORTRAN I/O
C            LOGIC(2) = IFORCE, IF .TRUE. A FORCING FUNCTION IS USED
C                                  .FALSE. NO FORCING FUNCTION
C            LOGIC(3) = IDISP, IF .TRUE. INITIAL DISPLACEMENTS
C                                 .FALSE. NO INITIAL DISPLACEMENTS
C            LOGIC(4) = IVEL, IF .TRUE. INITIAL VELOCITIES
C                                .FALSE. NO INITIAL VELOCITIES
C            LOGIC(5) = FIXSTP, IF .TRUE. USE A FIXED TIME STEP
C                                  .FALSE. VARIABLE STEP WILL BE USED
C            LOGIC(6) = IDAMP, IF .TRUE. PROBLEM HAS DAMPING
C                                 .FALSE. NO DAMPING
C            LOGIC(7) = IDMPDG, IF .TRUE. DAMPING MATRIX IS DIAGONAL
C                                  .FALSE. DAMPING MATRIX NONDIAGONAL
C
C        NOTE  IF IDMPDG = .TRUE., THEN IDAMP HAS THE FOLLOWING MEANING
C           1) IF IDAMP = .TRUE.  USER WILL SUPPLY DAMPING MATRIX AS
C                  A VECTOR, TO BE READ IN BY STINT
C           2) IF IDAMP = .FALSE.  DAMPING IS SUPPLIED BY A USER ROUTINE
C                  THAT COMPUTES D*UD AS IN THE NONDIAGONAL DAMPING CASE
C
C            INTGR(1) = MAXDEG, THE NUMBER OF DEGREES OF FREEDOM
C            INTGR(2) = IUNIT,  THE EXTERNAL MASS STORAGE UNIT NUMBER
C                           THAT CONTAINS THE MASS MATRIX, DAMPING
C                           MATRIX (IF IDMPDG AND IDAMP ARE .TRUE.),
C                           INITIAL DISPLACEMENT AND VELOCITY (IF
C                           PRESENT).  ALL QUANTITIES ARE VECTORS OF
C                           LENGTH MAXDEG.
C            INTGR(3) = KBAND, THE WIDTH OF THE CONNECTIVITY BAND OF
C                           THE STIFFNESS MATRIX (OPERATOR) OVER WHICH
C                           THE ERROR IS TO BE NORMALIZED.
C                           NOTE  KBAND = 1 IMPLIES EACH DOF IS TREATED
C                                 EQUALLY
C                                 KBAND = MAXDEG/10 + 2, RECOMMENDED
C                                 (WHERE MAXDEG/10 IS INTEGER ARITH-
C                                  METIC)
C
C            REALN(1) = TIME, THE STARTING VALUE OF TIME FOR THIS RUN
C            REALN(2) = TMAX, THE VALUE OF TIME AT WHICH THE INTEGRATOR
C                           WILL STOP
C            REALN(3) = DTMIN, THE MINIMUM TIME STEP
C            REALN(4) = DTMAX, THE MAXIMUM TIME STEP
C                           (NOTE, IF FIXSTP = .TRUE. STINTC WILL USE
C                           TIME STEP = DTMIN AND SET DTMAX=DTMIN)
C            REALN(5) = SAMPLES/CYCLE, THE NUMBER OF SAMPLES/CYCLE
C                           FOR THE DOMINANT FREQUENCY COMPONENT,
C                           MUST BE PI OR GREATER FOR STABILITY.
C                           10 TO 20 SAMPLES IS SUGGESTED
C                           4.0 IS DEFAULT, IF REALN(5) .LT. PI
C            REALN(6) = SAMPLES/CYCLE, THE NUMBER OF SAMPLES/CYCLE
C                           FOR THE HIGHEST APPARENT FREQUENCY
```

```
C                              COMPONENT, MUST BE PI OR GREATER
C                              FOR STABILITY.
C                              4PI IS INCREDIBILY STABLE
C                              4.0 IS DEFAULT, IF REALN(6) .LT. PI
C             REALN(7) = ALFA, WHERE THE VALUE OF ALFA LIES BETWEEN 0.0
C                        AND 1.0.  SUGGEST ALFA NEAR 1.0 FOR LIGHT DAMP
C                        ING AND NEAR 0.20 FOR CRITICAL DAMPING (SEE
C                        REFERENCES 3 & 4).  ONLY USED
C                        FOR NONDIAGONAL DAMPING
C
C             ICORE  IS THE SIZE OF THE DATA ARRAY C THAT STINTC NEEDS
C                    TO DO ITS THING.  THE USER MUST INSURE THAT THIS
C                    ALLOCATION IS AVAILABLE.
C
C             ICORE = (3 + 3*L1 + L2 + L3 + L4)*MAXDEG
C                 WHERE  L1 = 3 FOR VARIABLE STEP, 1 FOR FIXED STEP
C                        L2 = 1 FOR DAMPING, 0 OTHERWISE
C                        L3 = 1 FOR DIAGONAL DAMPING, 0 OTHERWISE
C                        L4 = 1 FOR VARIABLE STEP, 0 OTHERWISE
C
      LOGICAL LOGIC(20)
      INTEGER INTGR(20)
      REAL    REALN(20)
      COMMON  C(100)
C
      REAL SM(2),VZ(2)
C
      LOGIC(1) = .FALSE.                  ! UNFORMATTED FORTRAN I/O
      LOGIC(2) = .TRUE.                   ! FORCING FUNCTION
      LOGIC(3) = .FALSE.                  ! NO INITIAL DISPLACEMENT
      LOGIC(4) = .TRUE.                   ! INITIAL VELOCITY
      LOGIC(5) = .FALSE.                  ! VARIABLE TIME INCREMENT
      LOGIC(6) = .TRUE.                   ! DAMPING PRESENT
      LOGIC(7) = .FALSE.                  ! NONDIAGONAL DAMPING
C
      INTGR(1) = 2                        ! 2 D.O.F.
      INTGR(2) = 1                        ! DATA ON UNIT 1
      INTGR(3) = 1                        ! KBAND = 1
C
      REALN(1) = 0.0                      ! START TIME
      REALN(2) = 1.0                      ! FINISH TIME
      REALN(3) = 0.00001                  ! MINIMUM TIME INCREMENT
      REALN(4) = 1.0                      ! MAXIMUM TIME INCREMENT
      REALN(5) = 50.0                     ! SAMPLE/CYCLE DOMINANT FREQ.
      REALN(6) = 4.0                      ! SAMPLE/CYCLE HIGHEST FREQ.
      REALN(7) = 1.0                      ! ALFA - DAMPING PARAMETER
C
      VZ(1) = 100.0                       ! INITIAL VELOCITY D.O.F. 1
      VZ(2) = 0.0                         ! INITIAL VELOCITY D.O.F. 2
C
      SM(1) = 1.0                         ! MASS  D.O.F. 1
      SM(2) = 1.0                         ! MASS  D.O.F. 2
C
      OPEN(UNIT=1, FORM='UNFORMATTED', TYPE='SCRATCH')
      WRITE (1) (SM(I),I=1,2)             ! UNFORMATTED OUTPUT TO UNIT 1
```

```
      WRITE (1) (VZ(I),I=1,2)          ! UNFORMATTED OUTPUT TO UNIT 1
      REWIND 1
C
      CALL STINTC(LOGIC,INTGR,REALN,C)
C
      WRITE(6,1)
    1 FORMAT(////,'  *** THAT''S ALL FOLKS ***')
C
      END


      SUBROUTINE FORCE(MAXDEG,TIME,F)
C
C --- PURPOSE TO PROVIDE THE USER SUPPLIED FORCING FUNCTION, F
C
C --- INPUT        MAXDEG,TIME
C
C --- OUTPUT       F
C
C --- DECLARATIONS INTEGER MAXDEG
C                  REAL    TIME,F(1)
C
C --- DEFINITIONS
C              MAXDEG = NUMBER OF DEGREES OF FREEDON (LENGTH OF F ARRAY)
C
C              TIME = THE TIME AT WHICH THE FORCE IS TO BE CALCULATED
C
C              F = THE VECTOR CONTAINING THE FORCE AT TIME.  F IS OF
C                  LENGTH MAXDEG.
C
C --- EXAMPLE
C
C --- D.O.F. 2 SQUARE FORCE = 3000.  0.5<TIME<0.55
C --- FORCE = 0.0  OTHERWISE
C
      REAL F(1)
C
      F(1) = 0.0
      F(2) = 0.0
      IF(TIME .LT. 0.5)                GO TO 10
      IF(TIME .GT. 0.55)               GO TO 10
      F(2) = 3000.
C
   10 RETURN
      END


      SUBROUTINE SFORCE(MAXDEG,U,FS)
C
C --- PURPOSE TO PROVIDE THE USER SUPPLIED STIFFNESS FORCES, (FS),
C        I.E. (FS) = [K]*(U)
C
C --- INPUT        MAXDEG,U
```

CENTRAL DIFFERENCE TIME INTEGRATOR

```
C
C --- OUTPUT       FS
C
C --- DECLARATIONS INTEGER MAXDEG
C                  REAL U(1),FS(1)
C
C --- DEFINITIONS
C             MAXDEG = NUMBER OF DEGREES-OF-FREEDOM (LENGTH OF U AND
C                      ARRAYS)
C
C             U = THE VECTOR CONTAINING THE DISPLACEMENTS
C
C             FS = THE VECTOR CONTAINING THE STIFFNESS FORCES
C
C
C --- EXAMPLE
C
      REAL U(1),FS(1)
C
C --- SPRING RATE COEFFICIENTS
      DATA SK1/1000./,SKC/100./,SK2/1000./
C
C --- SPRING (STIFFNESS) FORCES FOR D.O.F. 1 AND D.O.F. 2
      FS(1) = SK1*TANH(U(1)) + SKC*(U(1)-U(2))
      FS(2) = SK2*SINH(U(2)) + SKC*(U(2)-U(1))
C
      RETURN
      END


      SUBROUTINE DFORCE(MAXDEG,UD,FD)
C
C --- PURPOSE TO PROVIDE THE USER SUPPLIED DAMPING FORCES, (FD),
C        I.E. (FD) = [D]*(UD)
C
C --- INPUT        MAXDEG,UD
C
C --- OUTPUT       FD
C
C --- DECLARATIONS INTEGER MAXDEG
C                  REAL UD(1),FD(1)
C
C --- DEFINITIONS
C             MAXDEG = NUMBER OF DEGREES-OF-FREEDOM (LENGTH OF UD AND
C                      ARRAYS)
C
C             UD = THE VECTOR CONTAINING THE VELOCITIES
C
C             FD = THE VECTOR CONTAINING THE DAMPING FORCES
C
C
C --- EXAMPLE
C
      REAL UD(1),FD(1)
```

```
      DATA D /5.0/                       ! DAMPING VALUE
C
      FD(1) = D*(UD(1) - UD(2))          ! D.O.F. 1  DAMPING FORCE
      FD(2) = D*(UD(2) - UD(1))          ! D.O.F. 2  DAMPING FORCE
C
      RETURN
      END


      SUBROUTINE OUTPUT(IPRT,MAXDEG,TIME,U,UD)
C
C --- PURPOSE TO SUPPLY THE DISPLACEMENT AND VELOCITY AT EACH TIME
C             STEP.  THE USER CAN THEN USE THIS DATA FOR PRINTING
C             OR PLOTTING OUTPUT
C
C --- INPUT         IPRT,MAXDEG,TIME,U,UD
C
C --- OUTPUT        NONE
C
C --- DECLARATIONS INTEGER IPRT,MAXDEG
C                  REAL    TIME,U(1),UD(1)
C
C --- DEFINITIONS
C             IPRT = -2  STINT HAS ERRORED OFF NO MORE OUTPUT
C                  = 0  AT ALL TIMES OTHER THAN BEGINNING TIME
C                  = 2  AT TIME = BEGINNING TIME
C                  = 10 AT THE LAST TIME (PROBLEM IS FINISHED)
C                  NOTE, THE IPRT=2 FLAG CAN BE USED TO INITIALIZE
C                        AND/OR SET UP PRINTING AND/OR PLOTTING
C                        BUFFERS,I/O,ETC.
C
C             MAXDEG = NUMBER OF DEGREES OF FREEDOM
C                      (LENGTH OF U AND UD ARRAYS)
C
C             TIME = THE VALUE OF THE CURRENT TIME
C
C             U = THE CURRENT DISPLACEMENT VECTOR (MAXDEG VALUES)
C
C             UD = THE CURRENT VELOCITY VECTOR (MAXDEG VALUES)
C
C --- EXAMPLE
C
      REAL U(1),UD(1)
      DATA ICONT/0/
C
      IF(IPRT .EQ. -2)                   GO TO 30
C --- OUTPUT RESPONSE TO UNIT 10 FOR POST PROCESSOR PLOTS
C     IF(IPRT .EQ. 2) OPEN(UNIT=10, FORM='UNFORMATTED', TYPE='SCRATCH')
C     WRITE(10) TIME,(U(I),I=1,MAXDEG),(UD(I),I=1,MAXDEG)
C --- PRINT INITIAL CONDITIONS
      IF(IPRT .EQ. 2) WRITE(6,1)
    1 FORMAT(1H1)
      IF(IPRT .EQ. 2)                    GO TO 10
      IF(IPRT .EQ. 10)                   GO TO 10
```

```
      ICONT = ICONT + 1
C --- PRINT AT EVERY 100-TH TIME INCREMENT
      IF(MOD(ICONT,100) .NE. 0)       GO TO 20
C
   10 WRITE(6,2) TIME,(I,U(I),UD(I),I=1,MAXDEG)
    2 FORMAT(/,
     1 '      TIME =',E10.3,/,
     2 '      D.O.F.     DISPLACEMENT    VELOCITY',/,
     3 (7X,I1,6X,E10.3,6X,E10.3))
      WRITE(6,3)
    3 FORMAT(/)
C
   20 RETURN
C --- ERROR EXIT
   30 WRITE(6,4)
    4 FORMAT(//,' ***** CENDIF HAS ERRORED OFF *****')
      STOP
      END
```

Appendix C

Sample Problem Results

```
WELCOME TO STINTC THE STAND-ALONE CENTRAL DIFFERENCE TIME INTEGRAT

              STINTC REQUIRES        28  WORDS OF STORAGE

                        6 DECEMBER 1979  VERSION


   TIME = 0.000E+00
   D.O.F.     DISPLACEMENT    VELOCITY
     1         0.000E+00      0.100E+03
     2         0.000E+00      0.000E+00


$$$  STEP SIZE INCREASED TO 0.1500E-03  AT 0.1100E-02
$$$  STEP SIZE INCREASED TO 0.2250E-03  AT 0.1850E-02
$$$  STEP SIZE INCREASED TO 0.3375E-03  AT 0.2975E-02
$$$  STEP SIZE INCREASED TO 0.5063E-03  AT 0.4663E-02
$$$  STEP SIZE INCREASED TO 0.7594E-03  AT 0.7194E-02
$$$  STEP SIZE INCREASED TO 0.1139E-02  AT 0.1099E-01
$$$  STEP SIZE INCREASED TO 0.1709E-02  AT 0.1669E-01
$$$  STEP SIZE INCREASED TO 0.2563E-02  AT 0.2523E-01
$$$  STEP SIZE INCREASED TO 0.3844E-02  AT 0.3804E-01
$$$  STEP SIZE INCREASED TO 0.5767E-02  AT 0.5727E-01
$$$  STEP SIZE INCREASED TO 0.8650E-02  AT 0.1149E+00

   TIME = 0.409E+00
```

D.O.F.	DISPLACEMENT	VELOCITY
1	0.739E+00	-0.516E+02
2	-0.295E+00	-0.862E+01

$$$ STEP SIZE DECREASED TO 0.6107E-02 AT 0.7723E+00
$$$ STEP SIZE INCREASED TO 0.8145E-02 AT 0.8273E+00
$$$ STEP SIZE DECREASED TO 0.5430E-02 AT 0.8517E+00
$$$ STEP SIZE INCREASED TO 0.7701E-02 AT 0.9169E+00

TIME = 0.100E+01

D.O.F.	DISPLACEMENT	VELOCITY
1	-0.199E-01	0.103E+02
2	0.271E+00	0.312E+02

AVERAGE TIME STEP	0.56818E-02
NUMBER OF STEP INCREASES	13
NUMBER OF STEP DECREASES	2
NUMBER OF FACTORIZATIONS	0
NUMBER OF TIME STEPS	177
NUMBER OF SOLUTIONS	179

DT OCCURRENCES IN THE RANGES INDICATED

FROM	10.0	TO	20.0	TIMES DTMIN, DT OCCURRENCES WAS	16
FROM	20.0	TO	30.0	TIMES DTMIN, DT OCCURRENCES WAS	5
FROM	30.0	TO	40.0	TIMES DTMIN, DT OCCURRENCES WAS	5
FROM	50.0	TO	60.0	TIMES DTMIN, DT OCCURRENCES WAS	5
FROM	70.0	TO	80.0	TIMES DTMIN, DT OCCURRENCES WAS	5
FROM	100.0	TO	200.0	TIMES DTMIN, DT OCCURRENCES WAS	10
FROM	200.0	TO	300.0	TIMES DTMIN, DT OCCURRENCES WAS	5
FROM	300.0	TO	400.0	TIMES DTMIN, DT OCCURRENCES WAS	5
FROM	500.0	TO	600.0	TIMES DTMIN, DT OCCURRENCES WAS	22
FROM	600.0	TO	700.0	TIMES DTMIN, DT OCCURRENCES WAS	10
FROM	700.0	TO	800.0	TIMES DTMIN, DT OCCURRENCES WAS	10
FROM	800.0	TO	900.0	TIMES DTMIN, DT OCCURRENCES WAS	79

*** THAT'S ALL FOLKS ***

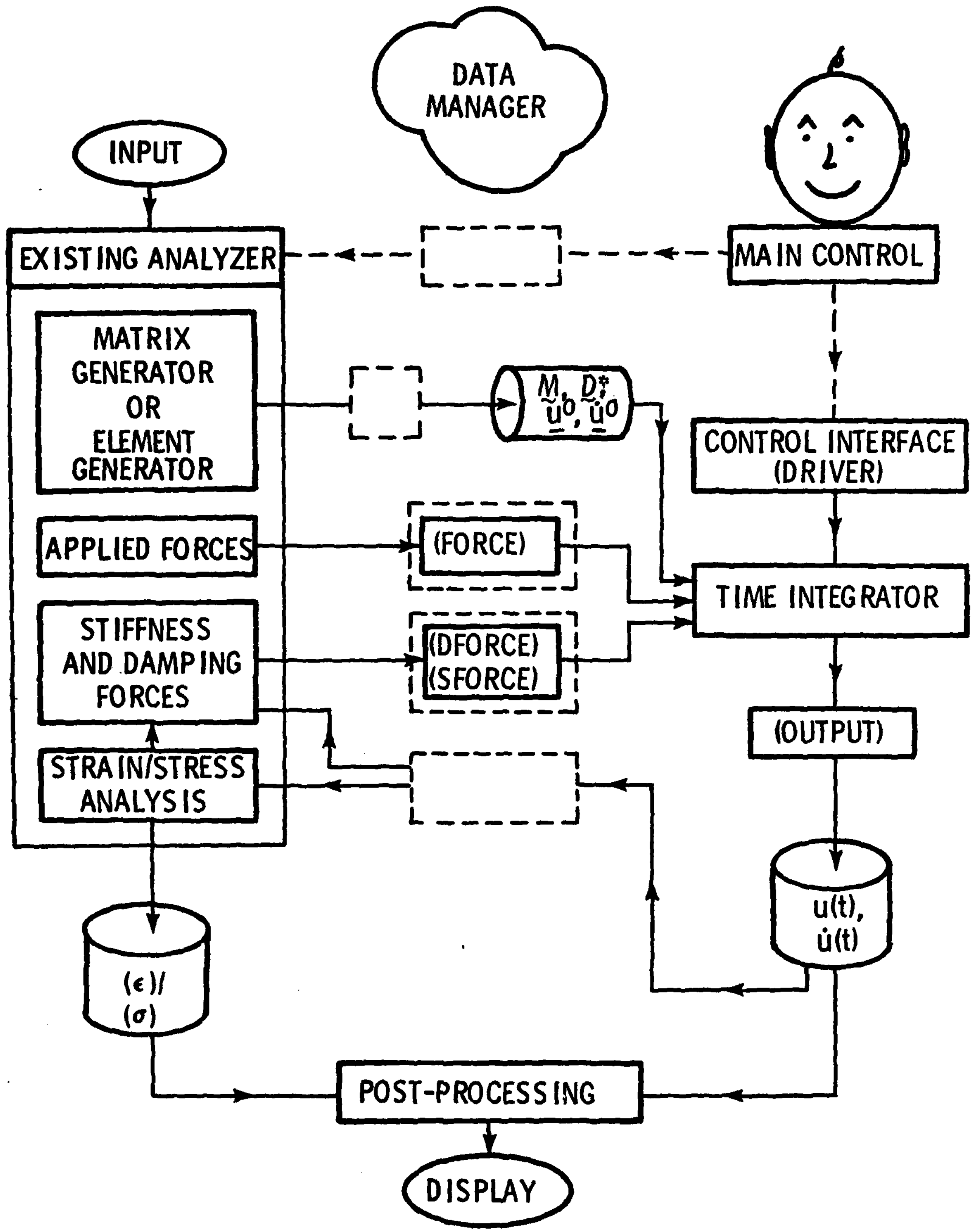


Figure 1. Overview of Structural Dynamic Response Analysis

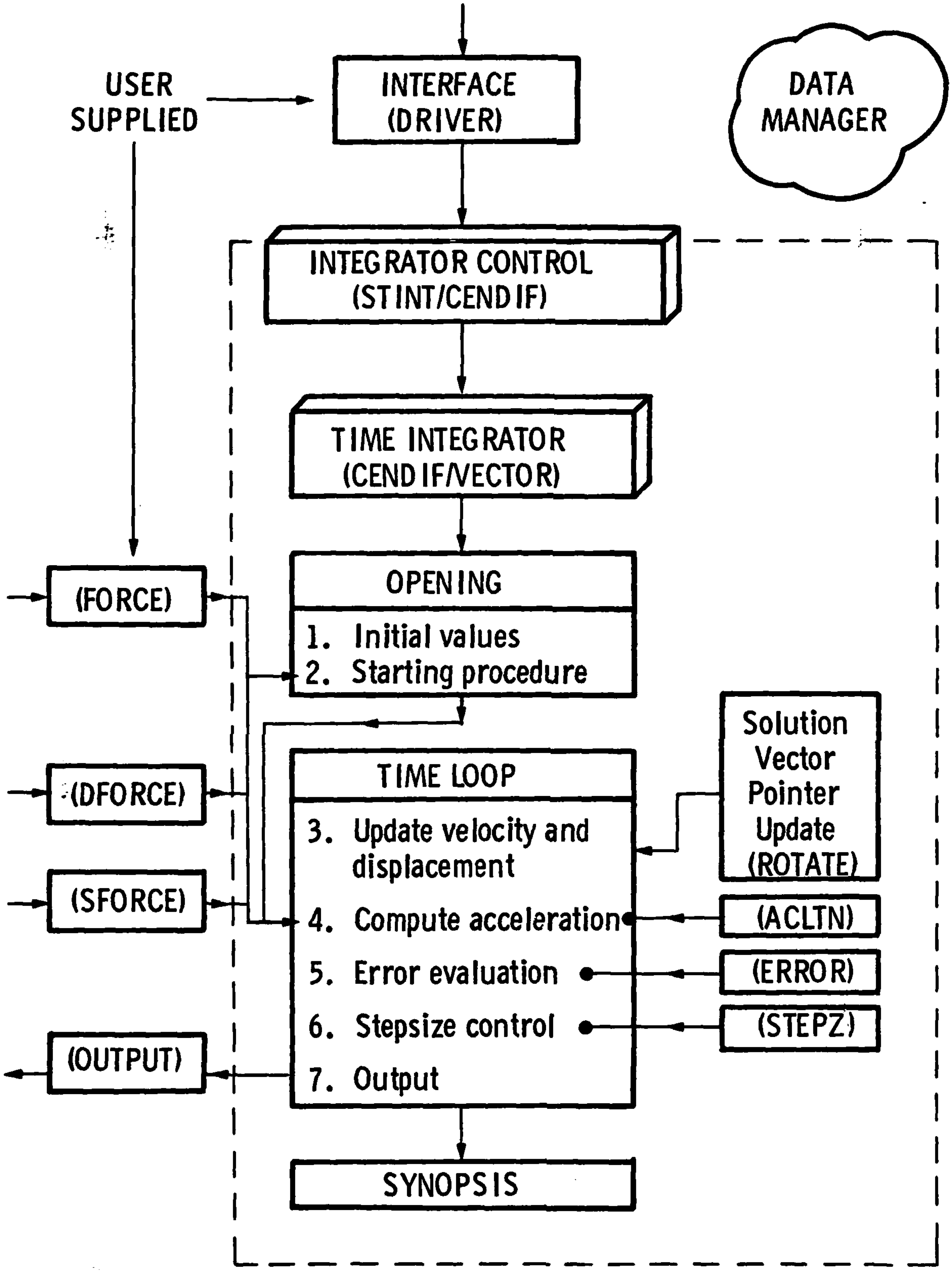
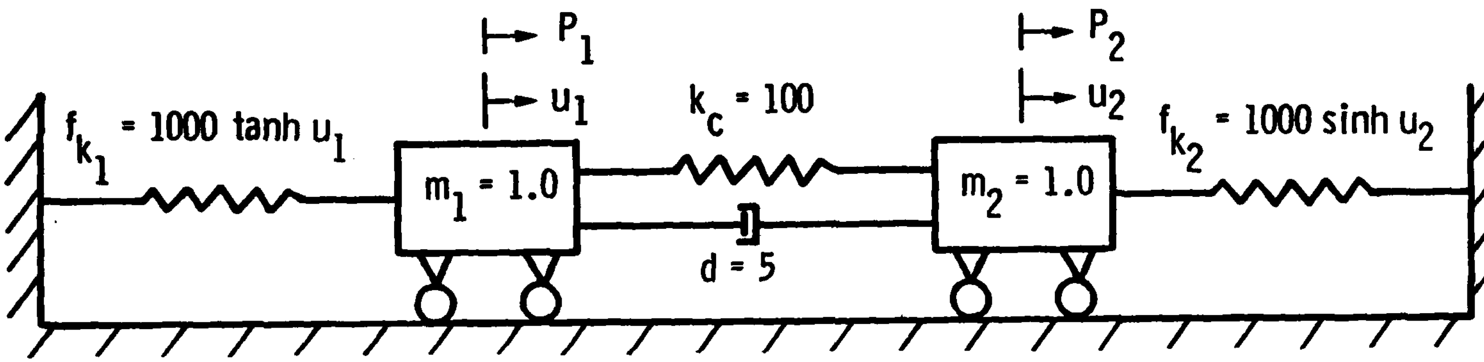


Figure 2. Time Integrator Functional Outline



$\dot{u}_1(0) = 100.0$ $\qquad$ $u_1(0) = u_2(0) = 0.0$

$\dot{u}_2(0) = 0.0$

$P_1(t) = 0.0$

$P_2(t) = 0.0 \qquad t < 0.5;\ t > 0.55$

$P_2(t) = 3000.0 \qquad 0.5 \leq t \leq 0.55$

Figure 3. Sample Problem Model

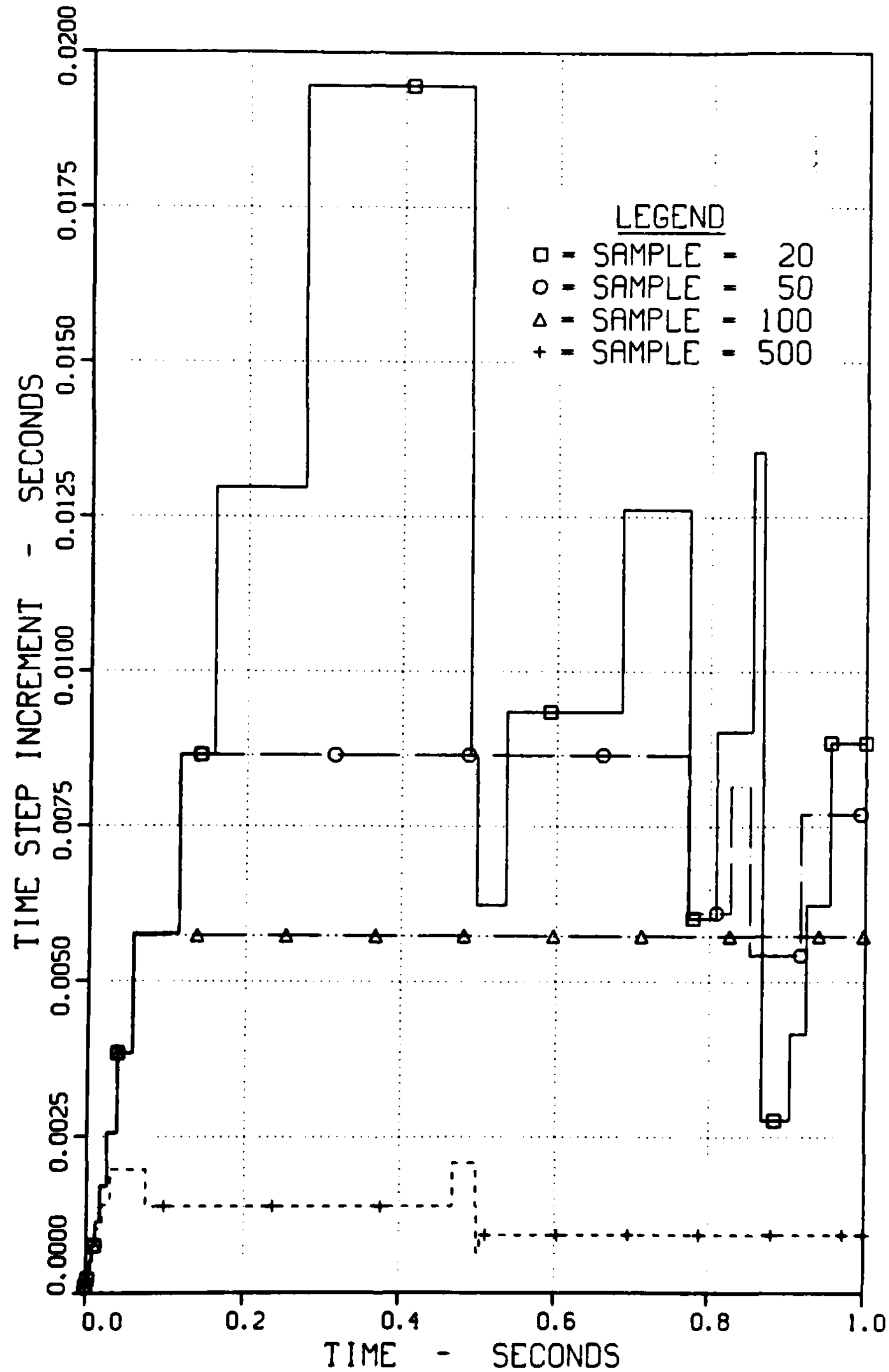


Figure 4. Sample Problem - Time Increment History

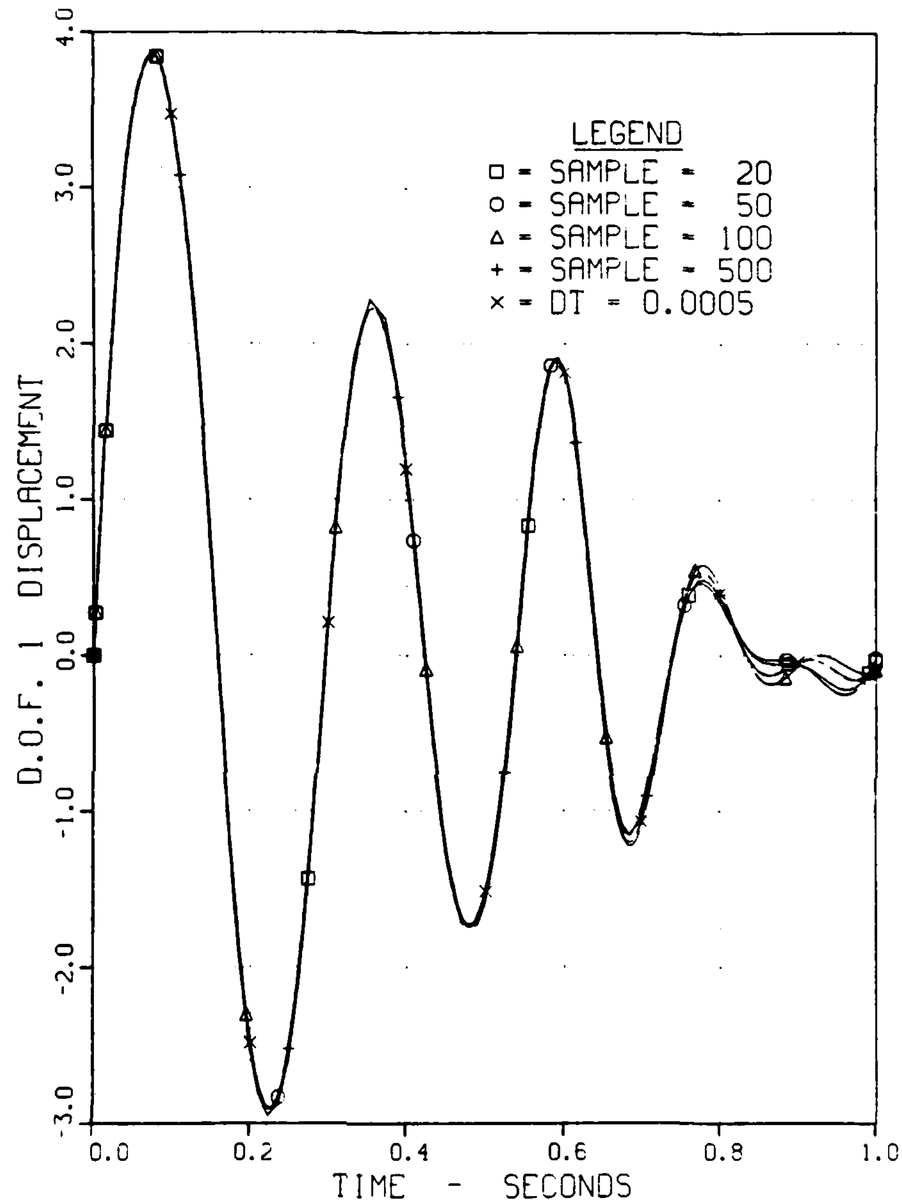


__Figure 5__. Sample Problem - D.O.F. 1 Displacement History

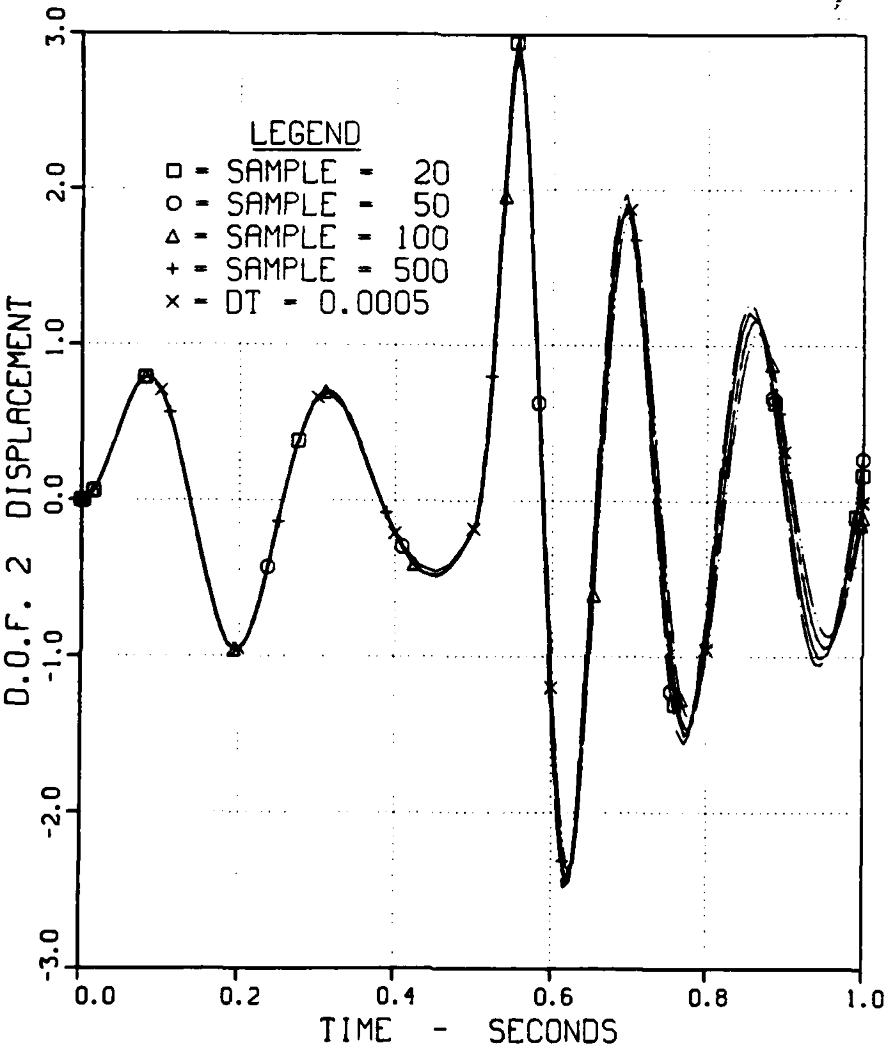


Figure 6. Sample Problem - D.O.F. 2 Displacement History

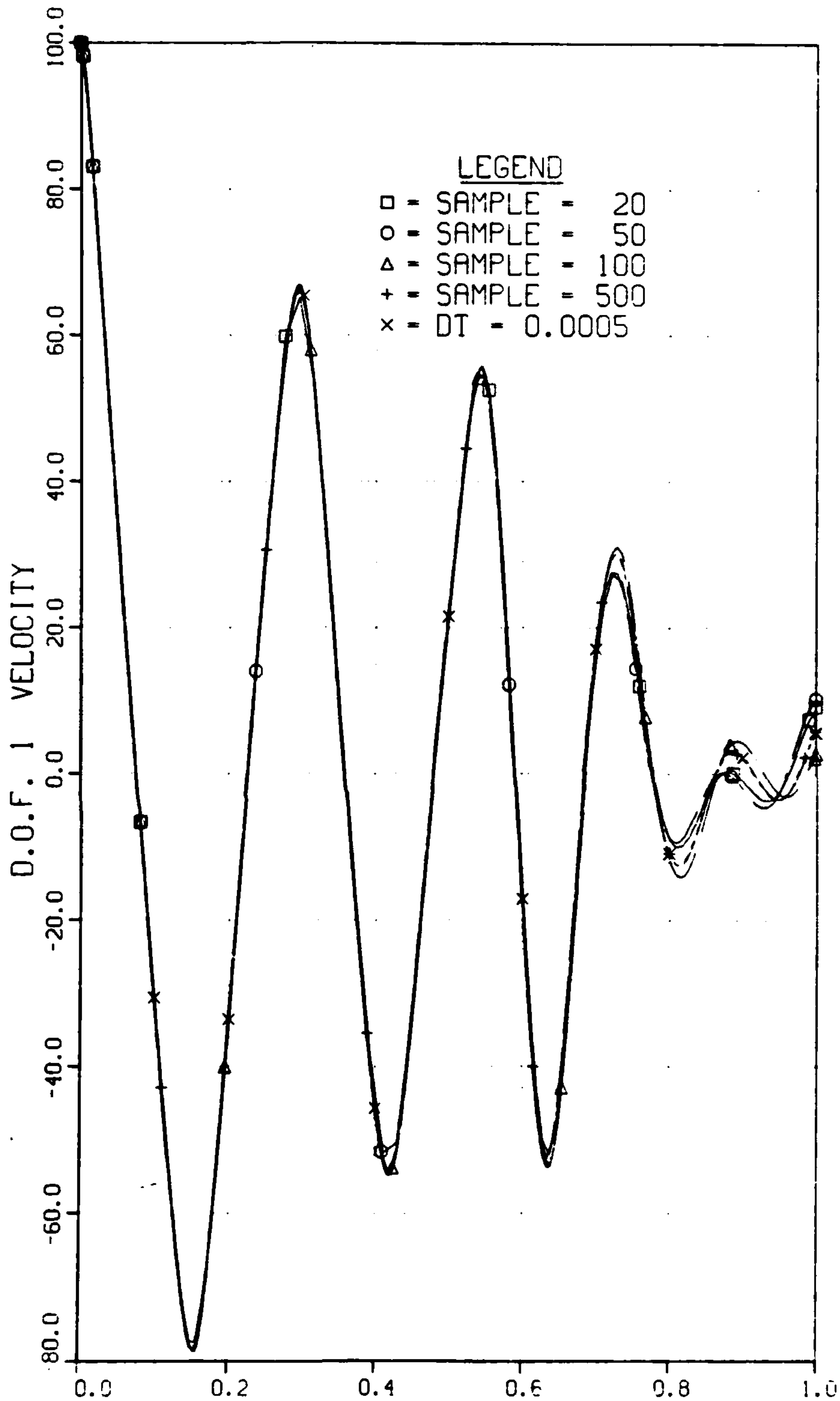


Figure 7. Sample Problem - D.O.F. 1 Velocity History

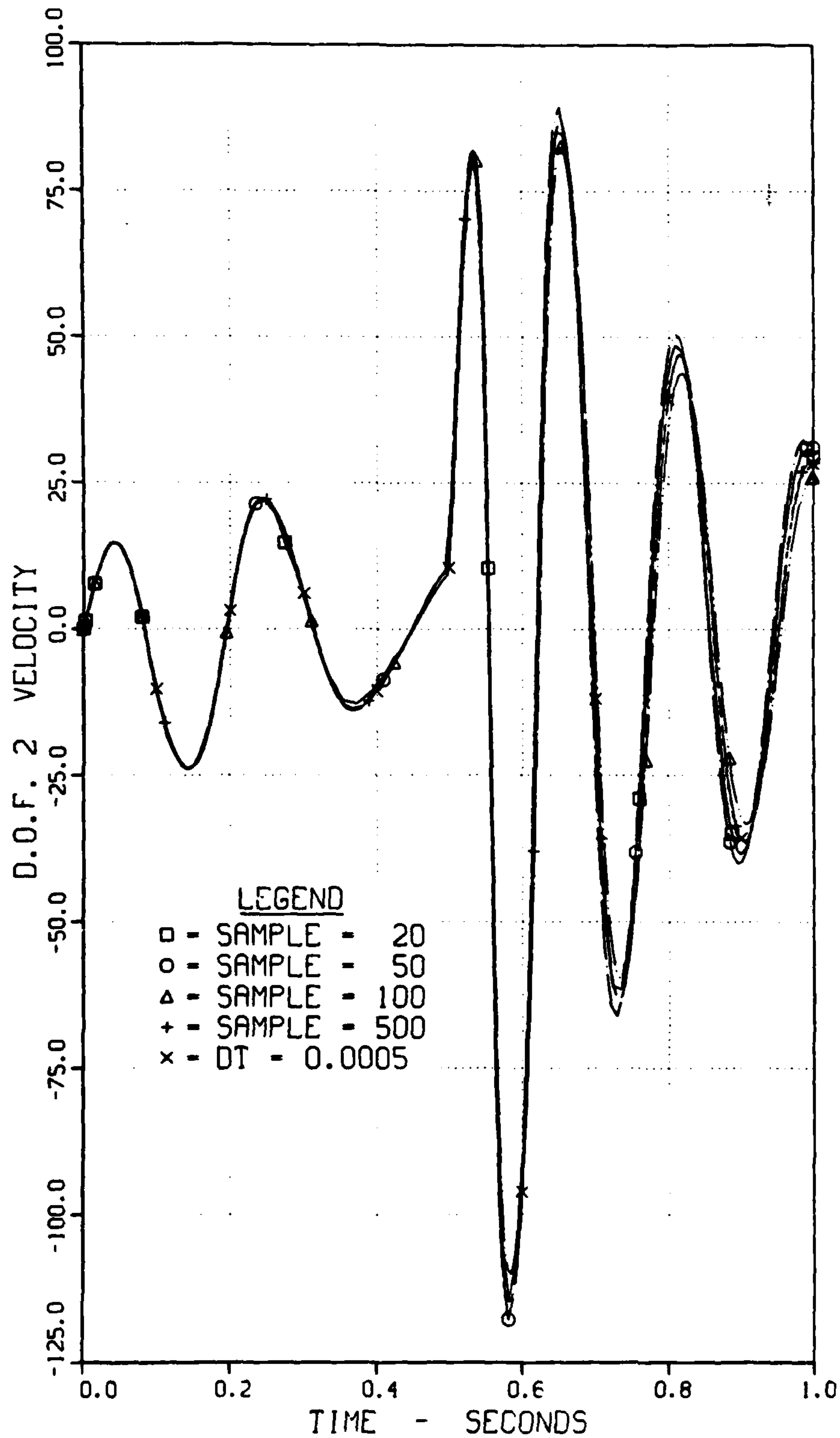


Figure 8. **Sample Problem - D.O.F. 2 Velocity History**

DAT
ILM